UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/726,272	12/01/2003	Satyanarayana Raju	50325-0847	3352

29989 7590 08/13/2007

HICKMAN PALERMO TRUONG & BECKER, LLP
2055 GATEWAY PLACE
SUITE 550
SAN JOSE, CA 95110

EXAMINER	
QIAN, SONGWEI	
ART UNIT	**PAPER NUMBER**
2109	

MAIL DATE	DELIVERY MODE
08/13/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

MN

Office Action Summary	Application No.	Applicant(s)	
	10/726,272	RAJU ET AL.	
	Examiner	Art Unit	
	Songwei Qian	2109	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *05 July 2007*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1, 5-8, 10-11, 13, 15-19, and 21-40* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1,5-8,10,11,13,15-19 and 21-40* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *01 December 2003* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-40 are pending in this application.

2. Claims 2-4, 9, 12, 14, 20 are cancelled by applicant on 07/05/2007.

3. Claims 31-40 are added by applicant on 07/05/2007.

4. Claims 1, 6-8, 10-11, 13, 15-19, 21, 23-25, 27-30 are amended by applicant on 07/05/2007.

5. Claims 1, 5-8, 10-11, 13, 15-19, and 21-40 are presented for examination.

Claim Objections

6. As for claims 15 and 21, the claims are objected to because of improper dependence as listed in the following.

a) Claim 15 is dependent on the cancelled claim 14. For examining purpose, claim 15 is considered to be dependent on claim 11.

b) Claim 21 is dependent on the cancelled claim 20. For examining purpose, claim 21 is considered to be dependent on claim 19.

Appropriate correction is required.

7. Claims 31-40 are objected to because of the following informalities: (paragraph **) at the end of each of these claims. (The attorney is thanked for providing paragraph

numbers to support the newly added claims. However, this information should be put in Applicant Arguments/Remarks.) Appropriate correction is required.

8. Claim 35 is objected to because of the following informalities: apparent typing error "thee", line 2. Appropriate correction is required.

Claim Rejections - 35 USC § 101

9. 35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

10. Claims 18-23 are rejected under 35 U.S.C. 101 as the claimed invention is directed to non-statutory subject matter.

11. As for claims 18-23, the claims fail to place the invention squarely within one statutory class of invention. In the specification, applicant has provided evidence that applicant intends the "computer-readable medium" to include "a carrier wave" ([0029], Line 5, and other places). As such, the claim is drawn to a form of energy. Energy is not one of the four categories of invention and therefore this claim is not statutory. Energy is not a series of steps or acts and thus is not a process. Energy is not a physical article or object and as such is not a machine or manufacture. Energy is not a combination of substances and therefor not a composition of matter.

Claim Rejections - 35 USC § 112

12. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

13. Claims 10, 15, 17, 21, and 30 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

14. As for claims 10, 15, 17, 21, 30, and 33 the phrase "the second software component" as listed in the following places, is not clearly understood and renders the claims indefinite. It is unclear what the phrase means, the existing version of the second software component that needs to be changed as cited in claims 1, 7, 11, 18, and 24, or the upgrade of the second software component. For examining purpose, the phrase "the second software component" listed in the following places is interpreted as "the upgrade of the second software component".

a) claim 10, Line 2;

b) claim 15, Line 2;

c) claim 17, Line 3;

d) claim 21, Lines 3-4;

e) claim 30, Line 3.

15. As for claim 33, the phrase "from a belonging to the network" renders the claims indefinite because it is unclear as to what the phrase means.

16. As for claim 35, the phrase "evaluate current states of the processing engine without considering any remnants from any prior instances of the processing engine" renders the claims indefinite because it is unclear as to what the phrase means.

Claim Rejections - 35 USC § 103

17. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

18. Claims 1, 5-8, 10-11, 13, 15-19, 21-30, and 32 are rejected under 35 U.S.C. 103(a) as being unpatentable over Mayer (US Pub. # 2002/0019864 A1) in view of Erickson et al. (US Pub. # 2003/0177223 A1), hereinafter "Erickson". Both of these two references are cited by the examiner in the previous office action.

19. As for claim 1, Mayer discloses:

A method for managing versions of a plurality of software components on a

network ([0017]), comprising:

detecting a version change to a first software component out of the plurality of the software components (determine any configuration changes of the managed elements , [0022], Lines 3-4);

Although Mayer discloses "a database containing the configuration of managed elements" (Fig. 6 and [0043]), Mayer does not appear to explicitly disclose:

automatically identifying a second software component out of the plurality of the software components, that needs to be changed to be compatible with the first software component, wherein the second software component depends on the first software component.

assessing the software dependencies and compatibilities of at least the first and second software components;

downloading upgrades of the first and second software component from a network device;

storing the upgrades of the first and second software components in a component cache; and

checking version information of the upgrades of the first and second software components stored in the component cache.

However, Erickson discloses:

automatically identifying a second software component out of the plurality of the software components, that needs to be changed to be compatible with the first software component, wherein the second software component depends on the first software component (Fig. 2 and [0019], Lines 1-5; note that "the computing device 110 may include a variety of devices having a programmable device, such as, but not limited to, servers, personal computers, PDAs, etc.", [0024], Lines 23-25).

assessing the software dependencies and compatibilities of at least the first and second software components (Fig. 2, [0019], Lines 1-5, and [0022]);

downloading upgrades of the first and second software component from a network device ([0017], Lines 9-10);

storing the upgrades of the first and second software components in a component cache (The downloaded version is stored in the memory 120, [0017], Lines 12-13); and

checking version information of the upgrades of the first and second software components stored in the component cache ([0017], Lines 1-5).

It would have been obvious to one of ordinary skill in the art at the time of invention was made to combine the teachings of Mayer with the teachings of Erickson by

automatically identifying a second software component out of the plurality of the software components, that needs to be changed to be compatible with the first software component, wherein the second software component depends on the first software component.

assessing the software dependencies and compatibilities of at least the first and second software components;

downloading upgrades of the first and second software component from a network device;

storing the upgrades of the first and second software components in a component cache; and

checking version information of the upgrades of the first and second software components stored in the component cache.

The motivation for combining the teachings of Mayer with the teachings of Erickson is to insure that the versions (managed elements) are compatible (Erickson, [0004], Lines 7-8) and the (network) system functions properly (Erickson, [0005], Line 5), to avoid unnecessary download of software updates, to avoid large network transport overhead (Mayer, [0012], Lines 4-5), to avoid excessive time required to perform software updates by the system administrator (Erickson, [0006], Lines 15-17), and to avoid mistakes made by the system administrator for updates (Erickson, [0006], Lines 11-12).

20. As for claim 7, Mayer discloses:

A method for managing versions of a plurality of software components on a network ([0017]), comprising:

detecting a version change to a first software component out of the plurality of the software components (determine any configuration changes of the managed elements , [0022], Lines 3-4);

Although Mayer discloses "a database containing the configuration of managed elements" (Fig. 6 and [0043]), Mayer does not appear to explicitly disclose:

automatically identifying a second software component out of the plurality of the software components that needs to be changed to be compatible with the first software component, wherein the second software component depends on the first software component;

assessing the software dependencies and compatibilities of at least the first and second upgrades of software components;

collecting attributes of the second soft-ware component; and

automatically manipulating the second software component according to the attributes, wherein the manipulating step further comprises downloading a copy of an upgrade of the second software component and storing it within a component cache.

However, Erickson discloses:

automatically identifying a second software component out of the plurality of the software components that needs to be changed to be compatible with the first software component, wherein the second software component depends on the first software component (Fig. 2 and [0019], Lines 1-5; note that "the computing device 110 may include a variety of devices having a programmable device, such as, but not limited to, servers, personal computers, PDAs, etc.", [0024], Lines 23-25);

assessing the software dependencies and compatibilities of at least the first and second upgrades of software components (Fig. 2, [0019], Lines 1-5, and [0022]);

collecting attributes of the second soft-ware component (firmware versions, [0016], Lines 1-2); and

automatically manipulating the second software component according to the attributes ([0027], Lines 7-11), wherein the manipulating step further comprises downloading a copy of an upgrade of the second software component ([0017], Lines 9-10 and [0027], Lines 7-11) and storing it within a component cache (The downloaded version is stored in the memory 120, [0017], Lines 12-13).

It would have been obvious to one of ordinary skill in the art at the time of invention was made to combine the teachings of Mayer with the teachings of Erickson by

automatically identifying a second software component out of the plurality of the software components that needs to be changed to be compatible with the first software component, wherein the second software component depends on the first software component;

assessing the software dependencies and compatibilities of at least the first and second upgrades of software components;

collecting attributes of the second soft-ware component; and

automatically manipulating the second software component according to the attributes, wherein the manipulating step further comprises downloading a copy of an upgrade of the second software component and storing it within a component cache.

The motivation for combining the teachings of Mayer with the teachings of Erickson is to insure that the versions (managed elements) are compatible (Erickson, [0004], Lines 7-8) and the (network) system functions properly (Erickson, [0005], Line 5), to avoid unnecessary download of software updates, to avoid large network transport overhead (Mayer, [0012], Lines 4-5), to avoid excessive time required to perform software updates by the system administrator (Erickson, [0006], Lines 15-17), and to avoid mistakes made by the system administrator for updates (Erickson, [0006], Lines 11-12).

21. As for claim 11, Mayer discloses:

An apparatus for managing versions of a plurality of software components on a network ([0017]), comprising:

a user interface (management interface, [0058], Lines 1-2); and

a processing engine (distributed managers 2 and agent means, Fig. 1, [0052], Line 2 and [0028], Line 1), coupled to the user interface, wherein the processing engine further comprises:

an event manager (intelligent agents, [0026], Line 5) that detects a version change to a first software component out of the plurality of the software components ([0026], Line 4 and [0022], Lines 3-4);

Mayer does not appear to explicitly disclose:

a component manager that in response obtains version information of first software component from a version manager and automatically identifies an upgrade to

a second software component out of the plurality of the software components that needs to be changed to be compatible with the first software component, wherein the second software component depends on the first software component;

wherein the component manager automatically downloads the upgrade of the second software component and stores a copy of the upgrade of the second software component in a component cache.

However, Erickson discloses:

a component manager (a network administrator console (NAC) 140 and a service processor 124, Fig. 1, [0015], Line 1 and [0013], Lines 6-7) that in response obtains version information of first software component from a version manager (version check utility 142, Fig. 1 and [0016], Line 1) and automatically identifies a second software component out of the plurality of the software components that needs to be changed to be compatible with the first software component, wherein the second software component depends on the first software component (Fig. 2 and [0019], Lines 1-5; note that "the computing device 110 may include a variety of devices having a programmable device, such as, but not limited to, servers, personal computers, PDAs, etc.", [0024], Lines 23-25);

wherein the component manager automatically downloads the upgrade of the second software component ([0017], Lines 9-10 and [0027], Lines 7-11) and stores a copy of the upgrade of the second software component in a component cache (The downloaded version is stored in the memory 120, [0017], Lines 12-13).

It would have been obvious to one of ordinary skill in the art at the time of invention was made to combine the teachings of Mayer with the teachings of Erickson by having

a component manager that in response obtains version information of first software component from a version manager and automatically identifies an upgrade to a second software component out of the plurality of the software components that needs to be changed to be compatible with the first software component, wherein the second software component depends on the first software component;

wherein the component manager automatically downloads the upgrade of the second software component and stores a copy of the upgrade of the second software component in a component cache.

The motivation for combining the teachings of Mayer with the teachings of Erickson is to insure that the versions (managed elements) are compatible (Erickson, [0004], Lines 7-8) and the (network) system functions properly (Erickson, [0005], Line 5).

22. As for claim 18, Mayer discloses:

detecting a version change to a first software component out of the plurality of the software components (determine any configuration changes of the managed elements , [0022], Lines 3-4);

Although Mayer discloses "a database containing the configuration of managed elements" (Fig. 6 and [0043]), Mayer does not appear to explicitly disclose:

automatically identifying a second software component out of the plurality of the software components, that needs to be changed to be compatible with the first software component, wherein the second software component depends on the first software component.

assess the software dependencies and compatibilities of at least the first and second upgrades of software components;

download a copy of an upgrade of the second software component; and

store a copy of the upgrade of the second software component in a component cache.

However, Erickson discloses:

automatically identifying a second software component out of the plurality of the software components, that needs to be changed to be compatible with the first software component, wherein the second software component depends on the first software component (Fig. 2 and [0019], Lines 1-5; note that "the computing device 110 may include a variety of devices having a programmable device, such as, but not limited to, servers, personal computers, PDAs, etc.", [0024], Lines 23-25);

assess the software dependencies and compatibilities of at least the first and second upgrades of software components (Fig. 2, [0019], Lines 1-5, and [0022]);

download a copy of an upgrade of the second software component ([0017], Lines 9-10 and [0027], Lines 7-11) ; and

store a copy of the upgrade of the second software component in a component cache (The downloaded version is stored in the memory 120, [0017], Lines 12-13).

It would have been obvious to one of ordinary skill in the art at the time of invention was made to combine the teachings of Mayer with the teachings of Erickson by

automatically identifying a second software component out of the plurality of the software components, that needs to be changed to be compatible with the first software component, wherein the second software component depends on the first software component.

assess the software dependencies and compatibilities of at least the first and second upgrades of software components;

download a copy of an upgrade of the second software component; and

store a copy of the upgrade of the second software component in a component cache.

The motivation for combining the teachings of Mayer with the teachings of Erickson is to insure that the versions (managed elements) are compatible (Erickson, [0004], Lines 7-8) and the (network) system functions properly (Erickson, [0005], Line 5), to avoid unnecessary download of software updates, to avoid large network transport overhead (Mayer, [0012], Lines 4-5), to avoid excessive time required to perform software updates by the system administrator (Erickson, [0006], Lines 15-17), and to avoid mistakes made by the system administrator for updates (Erickson, [0006], Lines 11-12).

23. As for claim 24, Mayer discloses:

a user interface means (management interface, Fig. 8 and [0058], Lines 1-2); and

a processing means (distributed managers 2 and agent means, Fig. 1, [0052], Line 2 and [0028], Line 1), coupled to the user interface, wherein the processing means further includes:

a detection means (distributed managers 2 and agents 3, Fig. 1 and [0052], Line 2) for detecting a version change to a first software component out of the plurality of the software components (determine any configuration changes of the managed elements , [0022], Lines 3-4);

Mayer does not appear to explicitly disclose:

a component manager means, stored within a component cache, for assessing the software dependencies and compatibilities for upgrades of the plurality of software components;

a compatibility verification means for automatically identifying a second software component out of the plurality of the software components that needs to be changed to be compatible with the first software component, wherein the second software component depends on the first software component.

However, Erickson discloses:

a component manager means (a network administrator console (NAC) 140 and a service processor 124, Fig. 1, [0015], Line 1 and [0013], Lines 6-7), stored within a component cache (FIG. 1), for assessing the software dependencies and compatibilities for upgrades of the plurality of software components (Fig. 2, [0019], Lines 1-5, and [0022]);

a compatibility verification means (a network administrator console (NAC) 140 and a service processor 124, Fig. 1, [0015], Line 1 and [0013], Lines 6-7) for automatically identifying a second software component out of the plurality of the software components that needs to be changed to be compatible with the first software component, wherein the second software component depends on the first software component (Fig. 2 and [0019], Lines 1-5).

It would have been obvious to one of ordinary skill in the art at the time of invention was made to combine the teachings of Mayer with the teachings of Erickson by using a component manager means, stored within a component cache, to assess the software dependencies and compatibilities for upgrades of the plurality of software components and using a compatibility verification means as taught by Erickson to automatically identify a second software component out of the plurality of the software components that needs to be changed to be compatible with the first software component, wherein the second software component depends on the first software component. The motivation for combining the teachings of Mayer with the teachings of Erickson is to insure that the versions (managed elements) are compatible (Erickson, [0004], Lines 7-

8) and the (network) system functions properly (Erickson, [0005], Line 5), to avoid unnecessary download of software updates, to avoid large network transport overhead (Mayer, [0012], Lines 4-5), to avoid excessive time required to perform software updates by the system administrator (Erickson, [0006], Lines 15-17), and to avoid mistakes made by the system administrator for updates (Erickson, [0006], Lines 11-12).

24. As for claims 5 and 22, the claims are rejected for the same reasons as for claims 1 and 18. In addition, Erickson discloses:

collecting attributes of the second software component (firmware versions, [0016], Lines 1-2); and

automatically manipulating the second software component according to the attributes ([0027], Lines 7-11).

25. As for claims 6, 8, 17, 23, and 30, the claims are rejected for the same reasons as for claims 1, 7, 11, 18, and 24. In addition, Erickson discloses:

downloading the upgrade of the second software component as part of performing an instance of the method ([0017], Lines 9-10 and [0027], Lines 7-11); and

replacing an existing version of the second software component with the upgrade of the second software component that has been downloaded in the same instance ([0017], Lines 11-12).

26. As for claims 10, 15, 21, and 28, the claims are rejected for the same reasons as for claims 7, 11, 18, and 24. In addition, Erickson discloses:

checking version information of the second software component that is stored in the cache to determine whether to perform the downloading step ([0017], Lines 1-5).

27. As for claims 13 and 26, the claims are rejected for the same reasons as for claims 11 and 24. In addition, Erickson discloses:

the component manager (a network administrator console (NAC) 140 and a service processor 124, Fig. 1, [0015], Line 1 and [0013], Lines 6-7) informs an operator of the apparatus of the upgrade of the second software component via the user interface (generate an alert (step 350), Fig. 3 and [0026], Line 2).

28. As for claims 16 and 29, the claims are rejected for the same reasons as for claims 11 and 24. In addition, Erickson discloses:

a desktop manager (a network administrator console (NAC) 140, Fig. 1 and [0015], Line 1), coupled to the component manager (a network administrator console (NAC) 140 and a service processor 124, Fig. 1, [0015], Line 1 and [0013], Lines 6-7), wherein the desktop manager

collects attributes of the second software component (firmware versions, [0016], Lines 1-2); and

manipulates the second software component according to the attributes ([0027], Lines 7-11).

29. As for claims 19 and 25, the claims are rejected for the same reasons as for claims 18 and 24. In addition, Erickson discloses:

instructions which, when executed by the one or more processors, cause the one or more processors to automatically download the copy of the upgrade of the second software component ([0017], Lines 9-10).

30. As for claim 27, the claim is rejected for the same reasons as for claim 24. In addition, Erickson discloses:

the compatibility verification means stores a copy of an upgrade of the second software component in a component cache (The downloaded version is stored in the memory 120, [0017], Lines 12-13).

31. As for claim 32, the claim is rejected for the same reasons as for claim 11. In addition, Erickson discloses:

the component manager (a network administrator console (NAC) 140 and a service processor 124, Fig. 1, [0015], Line 1 and [0013], Lines 6-7) is stored within the component cache (FIG. 1).

32. Claim 31 is rejected under 35 U.S.C. 103(a) as being unpatentable over Mayer in view of Erickson, and further in view of Branscomb et al. (US Pat. # 7,240,364 B1), hereinafter "Branscomb".

33. As for claim 31, both Mayer and Erickson do not appear to explicitly discloses:

the processing engine employs a metadata driven interface to communicate with the network devices.

However, Branscomb discloses:

the processing engine (NMS, Col. 63, line 38) employs a metadata driven interface to communicate with the network devices (Col. 63, lines 36-43, Col. 70, lines 3-10, and FIG. 13B).

It would have been obvious to one of ordinary skill in the art at the time of invention was made to combine the teachings of Mayer and Erickson with the teachings of Branscomb by having the processing engine that employs a metadata driven interface to communicate with the network devices in order to authenticate the identities of network devices within a telecommunications network and to allow the NMS to definitively link each set of attributes with the appropriate network device in a network (Branscomb, Col. 3, lines 8-10 and 15-16).

34. Claims 33-34 are rejected under 35 U.S.C. 103(a) as being unpatentable over Mayer in view of Erickson, and further in view of Allen et al. (US Pat. # 6,785,891 B1), hereinafter "Allen".

35. As for claim 33, both Mayer and Erickson do not appear to explicitly discloses:

a user activates the user interface via a launch applet.

However, Allen discloses:

a user activates the user interface via a launch applet (providing a primary graphical user interface via applets, Col. 10, lines 51-55 and FIG. 4).

It would have been obvious to one of ordinary skill in the art at the time of invention was made to combine the teachings of Mayer and Erickson with the teachings of Allen by activating the user interface via a launch applet in order to maintain the system and perform software updates (Erickson, [0006], lines 16-17).

36. As for claim 34, both Mayer and Erickson do not appear to explicitly discloses:

the launch applet originates from a belonging to the network (applets running within HTML pages loaded into the first frame F1, Col. 10, lines 51-55 and FIG. 4).

However, Allen discloses:

the launch applet originates from a belonging to the network (providing a primary graphical user interface via applets, Col. 10, lines 51-55 and FIG. 4).

It would have been obvious to one of ordinary skill in the art at the time of invention was made to combine the teachings of Mayer and Erickson with the teachings of Allen by

having the launch applet that originates from a belonging to the network in order to maintain the system and perform software updates (Erickson, [0006], lines 16-17).

37. Claim 35 is rejected under 35 U.S.C. 103(a) as being unpatentable over Mayer in view of Erickson, and further in view of Waldin et al. (US Pat. # 6,052,531), hereinafter "Waldin".

38. As for claim 35, Mayer discloses:

the processing engine (distributed managers 2 and agent means, Fig. 1, [0052], Line 2 and [0028], Line 1).

However, Mayer does not appear to explicitly disclose:

the desktop manager initializes the component manager so that thee component manager can evaluate current states of the processing engine without considering any remnants from any prior instances of the processing engine.

On the other hand, Erickson discloses:

the desktop manager (a network administrator console (NAC) 140, Fig. 1 and [0015], Line 1) initializes the component manager (a network administrator console (NAC) 140 and a service processor 124, Fig. 1, [0015], Line 1 and [0013], Lines 6-7).

It would have been obvious to one of ordinary skill in the art at the time of invention was made to combine the teachings of Mayer with the teachings of Erickson by initializing the component manager by the desktop manager in order to maintain the system and perform software updates (Erickson, [0006], lines 16-17).

However, both Mayer and Erickson do not appear to explicitly disclose:

the component manager can evaluate current states of the processing engine without considering any remnants from any prior instances of the processing engine.

On the other hand, Waldin discloses:

the component manager can evaluate current states of the processing engine without considering any remnants from any prior instances of the processing engine (The DeltaUpdater 126 determines 506 the current state of the application, Col. 9, lines 2-10).

It would have been obvious to one of ordinary skill in the art at the time of invention was made to combine the teachings of Mayer and Erickson with the teachings of Waldin by using the component manager to evaluate current states of the processing engine without considering any remnants from any prior instances of the processing engine in order to maintain the system and perform software updates (Erickson, [0006], lines 16-17).

39. Claims 36-37 and 39-40 are rejected under 35 U.S.C. 103(a) as being unpatentable over Mayer in view of Erickson, and further in view of Hunt et al. (US Pat. # 6,349,137 B1), hereinafter "Hunt".

40. As for claims 36 and 37, both Mayer and Erickson do not appear to explicitly discloses:

determining if the software upgrades are already loaded into the component cache.

However, Hunt discloses:

determining if the software upgrades are already loaded into the component cache (determine whether those applications are in cache, Col. 4, lines 24-26).

It would have been obvious to one of ordinary skill in the art at the time of invention was made to combine the teachings of Mayer and Erickson with the teachings of Hunt by determining if the software upgrades are already loaded into the component cache in order to maintain the system, to perform software updates (Erickson, [0006], lines 16-17), to avoid unnecessary download of software updates, and to avoid large network transport overhead (Mayer, [0012], Lines 4-5).

41. As for claims 39 and 40, both Mayer and Erickson do not appear to explicitly discloses:

determining if a specific software upgrade has already been loaded into the component cache.

However, Hunt discloses:

determining if a specific software upgrade has already been loaded into the component cache (determine whether those applications are in cache, Col. 4, lines 24-26).

It would have been obvious to one of ordinary skill in the art at the time of invention was made to combine the teachings of Mayer and Erickson with the teachings of Hunt by determining if a specific software upgrade has already been loaded into the component cache in order to maintain the system, to perform software updates (Erickson, [0006], lines 16-17), to avoid unnecessary download of software updates, and to avoid large network transport overhead (Mayer, [0012], Lines 4-5).

42. Claim 38 is rejected under 35 U.S.C. 103(a) as being unpatentable over Mayer in view of Erickson, and further in view of Budhiraja et al. (US Pub. # 2002/0144256 A1), hereinafter "Budhiraja".

43. As for claim 38, both Mayer and Erickson do not appear to explicitly discloses:

the replacing of the software upgrades is achieved via custom class loader mechanism.

However, Budhiraja discloses:

the replacing of the software upgrades is achieved via custom class loader mechanism (Fig. 7 and [0051]).

It would have been obvious to one of ordinary skill in the art at the time of invention was made to combine the teachings of Mayer and Erickson with the teachings of Hunt by determining if a specific software upgrade has already been loaded into the component cache in order to maintain the system and to perform software updates (Erickson, [0006], lines 16-17).

Response to Arguments

44. Regarding applicant's remark (paragraph 3 of Page 11) with respect to claim objection, the cancellation of claim 9 overcomes the previous claim objection.

45. Regarding applicant's remark (paragraph 4 of Page 11) with respect to 101 rejection, the amendment to claim 18 does not overcome the previous 101 rejection (see the above rejection) as the amendment does not specifically identify computer readable medium as computer readable storage medium that does not include a carrier wave.

46. Applicant's arguments filed on 07/05/2007 with respect to amended claims 1, 7, 11, and 18 (paragraph 7 of Page 11) have been considered but are not persuasive. However, the new grounds of rejection are presented because of applicant's amendment.

47. Regarding applicant's remark (paragraph 7 of Page 11) with respect to amended claims 1, 7, 11, and 18, examiner respectfully traverses applicant's assertion that Erickson does not teach the limitation "a component cache" cited in the amended claims 1, 7, 11, and 18. The amended claims 1, 7, 11, and 18 of the instant application explicitly claim "storing the upgrades of the first and second software components in a component cache" after the step of downloading the upgrades. "A component cache" here is used to store downloaded upgrades. Erickson teaches "the downloaded version is stored in the memory 120" ([0017], Lines 12-13"). So "the memory 120" here is used to store downloaded upgrades. Therefore, Erickson's "the memory 120" performs the same function as applicant's "a component cache" cited in applicant's amended claims 1, 7, 11, and 18.

48. Applicant's arguments filed on 07/05/2007 with respect to amended claims 1, 7, 18, and 24 (paragraph 8 of Page 11 to paragraph 2 of Page 12) have been considered but are not persuasive. However, the new grounds of rejection are presented because of applicant's amendment.

49. Regarding applicant's remark (paragraph 8 of Page 11 to paragraph 1 of Page 12) with respect to amended claims 1, 7, 18, and 24, examiner respectfully traverses applicant's assertion that Erickson's "recipe table 144" does not teach the limitation "software dependencies and compatibilities" cited in the amended claims 1, 7, 18, and 24. Erickson teaches ([0019]):

> The recipe table 144 includes a column 210 for the recipe name/version and columns 220 for the versions of firmware (F1 . . . Fn) that are dependent on each other and used together in the computing device 110. The table 144, for example, includes a row 230 for the recipe 10. The row 230 includes the versions V8, V6 and V4 for the firmware F1, F2 and Fn respectively. The row 240 for the recipe 1 may include the first versions for the firmware F1, F2 and Fn. Recipes may be published by the image server 130 after testing the versions of firmware for compatibility. The published recipe may then be downloaded by the NAC 140 or other devices that coordinate downloads of the updates 132.

Therefore, Erickson's "recipe table 144" does teach the limitation "software dependencies and compatibilities" cited in the amended claims 1, 7, 18, and 24

50. Applicant's other arguments with respect to claims 1-30 have been considered but are moot in view of the new ground(s) of rejection.

Conclusion

51. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

52. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Songwei Qian whose telephone number is 571-270-1910. The examiner can normally be reached on M-F (alternative Friday off 8:00am thru 5:00pm).

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Nabil El-Hady can be reached on 571-272-3963. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

SQ
08/09/2007

N. El-Hady

NABIL M. EL-HADY
SUPERVISORY PATENT EXAMINER